PREDICTIVE PERFORMANCE AND SCALABILITY MODELING OF A LARGE-SCALE APPLICATION

D.J. Kerbyson*, H.J. Alme*, A. Hoisie*, F. Petrini*, H.J. Wasserman*, M. Gittings**

* Los Alamos National Laboratory,
Parallel Architectures and Performance Team
CCS-3 Modeling, Algorithms and Informatics Group,
, Los Alamos, NM 87545
+1 (505)-667-4913
{djk,hoisie}@lanl.gov

**SAIC and
Los Alamos National Laboratory

ABSTRACT

In this work we present a predictive analytical model that encompasses the performance and scaling characteristics of an important ASCI application. SAGE (SAIC's Adaptive Grid Eulerian hydrocode) is a multidimensional hydrodynamics code with adaptive mesh refinement. The model is validated against measurements on several systems including ASCI Blue Mountain, ASCI White, and a Compaq Alphaserver ES45 system showing high accuracy. It is parametric - basic machine performance numbers (latency, MFLOPS rate, bandwidth) and application characteristics (problem size, decomposition method, etc.) serve as input. The model is applied to add insight into the performance of current systems, to reveal bottlenecks, and to illustrate where tuning efforts can be effective. We also use the model to predict performance on future systems.

Keywords

Performance analysis, full application codes, parallel system architecture, Teraflop scale computing.

1. INTRODUCTION

SAGE (SAIC's Adaptive Grid Eulerian hydrocode) is a multidimensional (1D, 2D, and 3D), multimaterial, Eulerian hydrodynamics code with adaptive mesh refinement (AMR). The code uses second order accurate numerical techniques. SAGE comes from the Los Alamos National Laboratory Crestone project, whose goal is the investigation of continuous adaptive Eulerian techniques to stockpile stewardship problems. SAGE has also been applied to a variety of problems in many areas of science and engineering including water shock, energy coupling, cratering and ground shock, stemming and containment, early time front end design, explosively generated air blast, and hydrodynamic instability problems [9]. SAGE represents a large class of production ASCI applications at Los Alamos that routinely run on 2000-4000 processors for months at a time.

SAGE is a large-scale parallel code written in Fortran 90, using MPI for inter-processor communications. Early versions of SAGE were developed for vector architectures. More recently, optimized versions of SAGE have been ported to all teraflop-scale ASCI architectures, as well as the CRAY T3E and Linux-based cluster systems.

This work describes a performance and scalability analysis of SAGE. One essential result is the development of a performance model that encapsulates the code's crucial performance and scaling characteristics. The performance model has been formulated from an analysis of the code, inspection of key data structures, and analysis of traces gathered at run-time. The model has been validated against a number of ASCI machines with high accuracy. The model is also applied in this work to predict the performance of SAGE on extreme-scale future architectures, such as clusters of SMPs. Included is the application of the model for predicting the performance of the code when algorithmic changes are implemented, such as using a different parallel data decomposition.

There are few existing performance studies that extend to full codes (for instance [10]), many tend to consider smaller applications especially in distributed environments (e.g. [5,7]). This paper represents an example of performance engineering applied to a full-blown code. SAGE has been analyzed and a performance model proposed, and validated, on all architectures of interest. The validated model is utilized for point-design studies involving changes in the architectures on which the code is running and in the algorithms utilized in the code. A predictive performance model of another important ASCI application is described in previous work [4].

SC2001 November 2001, Denver (c) 2001 ACM 1-58113-293-X/01/0011 $5.00

2. ESSENTIAL CHARACTERISTICS OF SAGE

In this section the important characteristics of SAGE that affect its performance and scaling behavior are described. In particular, the spatial data decomposition, the scaling of the sub-grid, the common operations within a code cycle and the adaptive mesh refinement operations are analyzed. In this work it is assumed that the spatial grid is three dimensional.

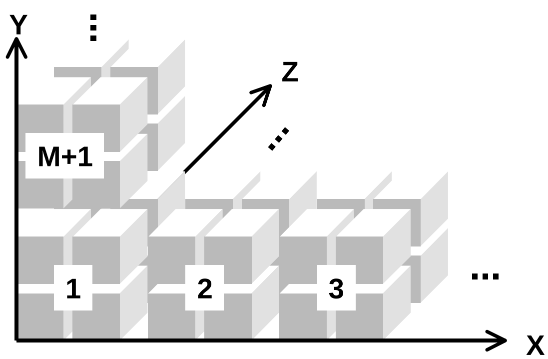
2.1 Parallel Spatial Decomposition in SAGE

SAGE uses a spatial discretization of the physical domain utilizing Cartesian grids. This spatial domain is partitioned across processors in “sub-grids” such that the first processor is assigned the first E cells in the grid (indexed in dimension order – X,Y,Z), the second processor is assigned the next E cells and so on. The assignment is actually done in blocks of 2x2x2 as illustrated in Figure 1a), where M is the number of blocks in the X-dimension. Figure 1b) illustrates the approximate assignment of cells over 4 processors (PEs). Note that the grid is primarily partitioned in the Z dimension. Each processor contains cells which are either:

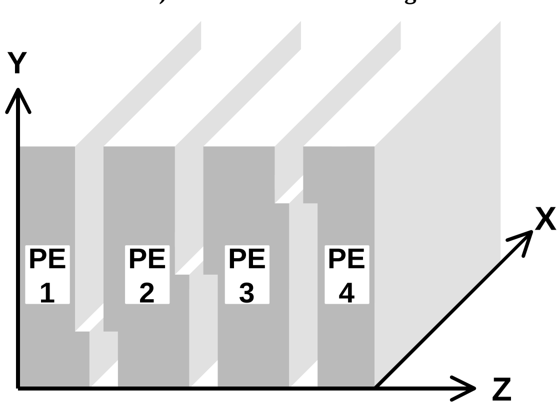
a) internal – all neighbor cells are contained on the same PE,

b) boundary – belong to one of the spatial domain’s physical boundaries (“faces”), or

c) inter-processor boundary – neighbor cells in physical space belong to sub-grids contained on different PEs (in one or more dimension).



a) Cell and block ordering



b) Example assignment of the spatial grid across four PEs

Figure 1. Cell and block assignment to Processors in SAGE.

A library designed for the communication requirements of the code is used to handle the necessary communications within SAGE. This includes the common MPI operations of allreduce and broadcast for instance, as well as two main application specific communication kernels: gather (get data) and scatter (put data) operations. These operations are used when processors require an update of their sub-grid with local cell information and inter-processor boundary data. The library uses a notion of *tokens* to record where all the necessary data can be found for each individual processor. A token is defined on each processor for each of cell centered values and cell face values in each data neighbor direction - 6 in total, and for the relationships of cells between levels in the AMR operation. Each token contains information on:

- sub-grid boundaries,
- data held locally within a processor,
- data held off processor (requiring communication), and
- data required off processor (also requiring communication).

2.2. Scaling of the Sub-grid

The sub-grid volume on each PE is a function of the number of cells per PE, a parameter which is specified in the input deck. SAGE assigns this number of cells to each PE. We are concerned with “weak scaling” in this analysis, in which the problem size grows proportionally with the number of processors resulting in each PE doing approximately the same amount of work.

The decomposition of the spatial grid across PEs is done in “slabs” (2-D partitioning), as suggested in Figure 1b). Each slab is uniquely assigned to one processor.

Taking the number of cells in each sub-grid to be E, the total grid volume V in cells is:

$$V = E.P = L^3 \tag{1}$$

and the volume of each sub-grid is:

$$E = l.L^2 \tag{2}$$

where P is the number of PEs, l is the short side of the slab (in the Z dimension) and L the side of the slab in X and Y directions (assuming a square grid in the X-Y plane). The surface of the slab, L^2, in the X-Y plane is:

$$L^2 = V^{2/3} = (E.P)^{2/3} \tag{3}$$

From this it can be seen that the surface increases as $P^{2/3}$. This sub-grid surface is directly proportional to the maximum data size that is communicated between PEs on data gather and scatter operations.

The maximum size of this surface that a PE will contain is constrained by E. In fact, since the assignment of cells to PEs is done in 2x2x2 blocks, the maximum surface is $E/2$, at which point the slab degenerates to a “foil” with a thickness of 2 cells. It is possible for the surface of the full spatial domain to be greater than E - thus resulting in each surface being assigned to more than one PE. This leads to physically neighboring data cells assigned to logically distant PEs. Hence communications will take place between more distant processors. The total communication requirements will remain as $(E.P)^{2/3}$, but will be dealt with by more than one PE.

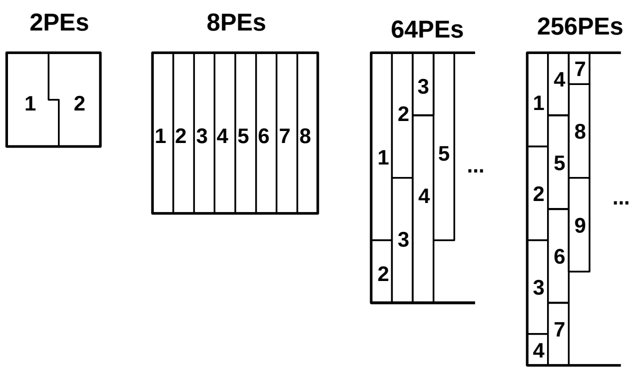


Figure 2. Cross-sections of the spatial grid and the assignment of cells to 2, 8, 64, and 256 processors.

Examples of the partitioning of the spatial grid across processors using slab decomposition are shown in Figure 2. The cross-section in the Z dimension is shown and it is assumed that E=13,500 cells throughout this work. Note that the volume of the spatial grid scales in accordance with equation 1. It can be seen that when using 2 and 8 processors, each PE holds more than one foil. In the case of 64 PEs each foil is mapped to two processors, and in the case of 256 PEs, each foil is held by 4 processors. The maximum logical distance between PEs on foil boundaries for the four cases shown is 1, 1, 2 and 4 respectively.

Consider again the volume of the entire grid:

$$V = E.P = (l.L^2).P \tag{4}$$

This is partitioned across PEs such that there will be $L/(2P)$ foils of width 2 on each PE, or:

$$(E.P)^{1/3}/2P \ = \ (E/8P^2)^{1/3} \tag{5}$$

When this has a value less than one, a processor will contain less than a single foil, i.e. when

$$P > SQRT(E/8) \tag{6}$$

The maximum distance between the processors that hold a foil, termed the "PE Distance" (*PED*) here is:

$$PED = \left\lceil \left(E/8P^2\right)^{-1/3} \right\rceil \tag{7}$$

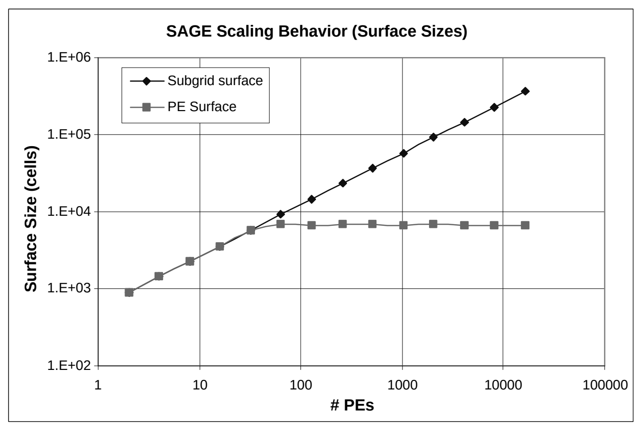
where $\lceil \ \rceil$ indicates an integer ceiling function. The minimum distance between the processors that hold that foil is:

$$MAX\left(\left\lceil \left(E/8P^2\right)^{-1/3} \right\rceil -1, 1\right) \tag{8}$$

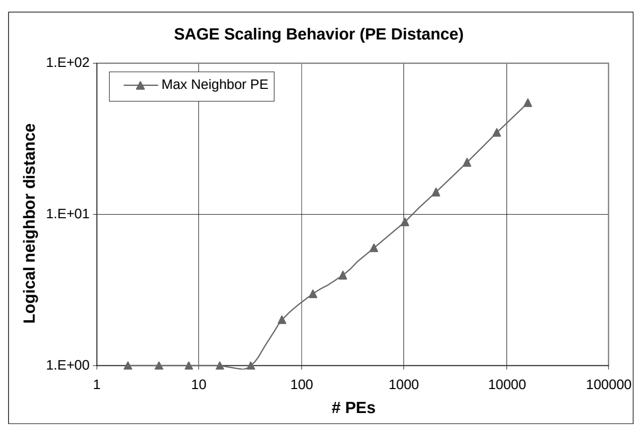
Thus when a processor is not assigned a full foil of the spatial domain, boundary exchange will involve all the processors that own the boundary, located at a logical distance *PED* apart.

The sub-grid surface, L^2, the actual inter-processor boundary area owned by a processor due to the slab degenerating to a foil and the subsequent splitting of the boundary amongst the processors within the *PED*, the "PE surface", and the *PED* are shown in Figure 3. It can be seen that the PE surface achieves a maximum after 32 PEs. The sub-grid surface approximately equals the PE surface multiplied by the *PED*. It is important to note that the *PED* is related to the communication requirements of the code, more precisely it is proportional to the size of messages generated in order to satisfy each necessary inter-processor boundary exchange. This is a consequence of the slab decomposition and could lead to communication inefficiencies depending on a specific machine topology.

A further observation related to the communication pattern is that when processors are considered to be labeled in a vector-like manner, from *0* to *P-1*, and with P_{SMP} processors per SMP box, out-of-box communication involves no more than a number of pairs of processors equal to the $min(PED, P_{SMP})$. Of course, if the *PED* is larger than P_{SMP}, more than two SMP boxes will be involved in the boundary exchange. As an example, on the ASCI Blue Mountain at Los Alamos, composed of SGI Origin 2000 boxes, given that P_{SMP} =128 and that, from Figure 2b) the PED is not larger than 100 for a reasonable number of PEs, no more than 2 Origin 2000 boxes will be involved in the communication for one boundary exchange.



a) sub-grid surface and PE surface sizes,



b) PED, the logical neighbor distance.

Figure 3. SAGE scaling characteristics

2.3 An Iteration *Cycle* of SAGE

The processing stages within a cycle typically involve three operations which are repeated a number of times dependent on the time interval utilized for integration of the equations in the code:

i) one (or more) gather operations to obtain a copy of the local and remote neighbor data

ii) computation in each of the local cells

iii) one (or more) scatter operations to update data on remote processors.

These three operations of SAGE directly relate to the surface-to-volume ratio of the code [2]. The first and the third stage define the surface, related to the amount and pattern of communication, while the second stage represents the volume, related to the amount of computation. The gather and scatter operations are performed using the token library described in section 2.1.

These three operations in a cycle of SAGE are shown schematically in Figure 4. In this example, it is assumed that the number of PEs (P) is 256, and the number of cells per PE (E) is 13,500. A single gather operation in all dimensions is depicted, followed by a processing step, and then a single scatter operation in all dimensions. The communication is shown only for processor n but in reality all processors perform communications of the same sizes, in the same direction, at the same time. In this example it can be seen that the main communication is in the Z dimension dealing with the sub-grid surface. The preponderance of communication in the Z dimension is also a consequence of the slab decomposition and is intuitive from Figure 1 b). The message sizes in both directions (HI and LO in SAGE terminology) of the three dimensions is shown in the box on the right side of Figure 4.

In addition to the gather and scatter operations, a number of other communications take place including several MPI type allreduce communications per cycle. A number of broadcast operations also exist but only during the initialization phase of the code.

The frequency of the gather/scatter operations was analyzed using MPI trace data. From this, the number of scatter/gather operations was taken to be 160 real and 17 integer operations per cycle. The surface communications in the Z -dimension represent 20% of the total number of messages but over 95% of the total communication time. In addition, 120 allreduce operations take place per cycle each of 4 bytes in size.

2.4 Adaptive Mesh Refinement in SAGE

SAGE performs adaptive mesh refinement operations at the end of each cycle iteration. Each cell in the spatial grid at this point may either be:

- split into a 2x2x2 block of cells, or
- combined with its neighbors, within the same cell block, to form a single cell, or
- remain unchanged.

The decision on whether to split or combine cells is determined by the current cell values in the calculation being performed. AMR enables more refined calculations to take place in those areas of the spatial grid characterized by more intense physical phenomena.

For example, the shock-wave indicated in the 2-D example in Figure 5 by the solid line may cause the cells associated with it (and close to it) to be split into smaller cells. In this example, cells are represented at a certain level of refinement. A cell at level 0 is not refined while a cell at level n represents a domain 8^n times smaller than one at level 0 in 3 dimensions.

The adaptive refinement of cells can result in load imbalance across processors, for instance when there is a large degree of activity in a localized region of the spatial grid in comparison to the grid as a whole. To overcome this, a load-balancing operation is performed at the end of each cycle when the maximum number of cells on any processor is greater than 10% above the average number of cells on the processors, i.e. load-balance if

$$\underset{i}{MAX}(E_i) > 1.1\left(\frac{1}{P}\sum_{i=1}^{P} E_i\right) \qquad (9)$$

where E_i is the number of cells on processor i.

The load-balancing operation takes advantage of the fact that the cells are organized into a one dimensional logical vector. The cells at level 0 are indexed in X, Y, Z ordering corresponding to positions in the spatial grid. By partitioning this vector into approximately equally sized segments, the number of cells can remain approximately equal among processors. However, the load-balancing process requires the communication of all data values associated with cells to be moved between processors. This can impact the overall application performance.

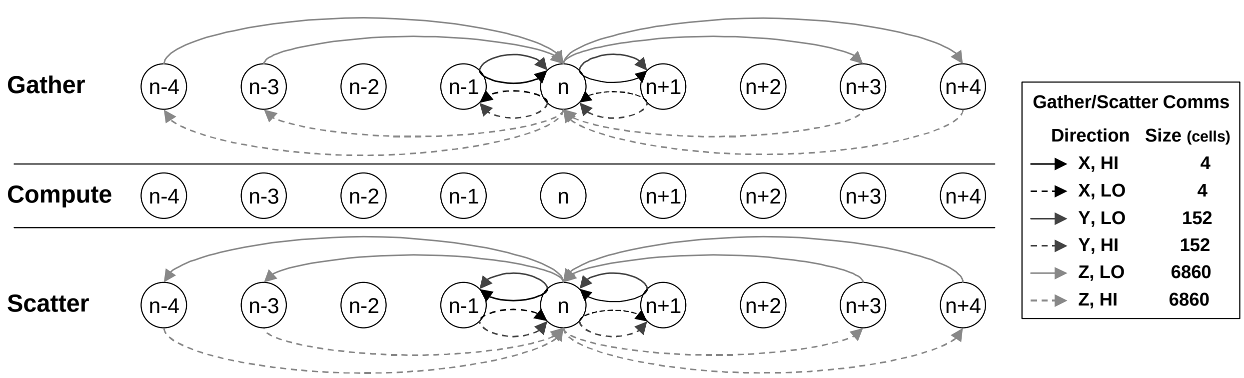


Figure 4. Schematic representation of the communication and computation within a cycle of SAGE consisting of: a data gather, processing, and a data scatter.

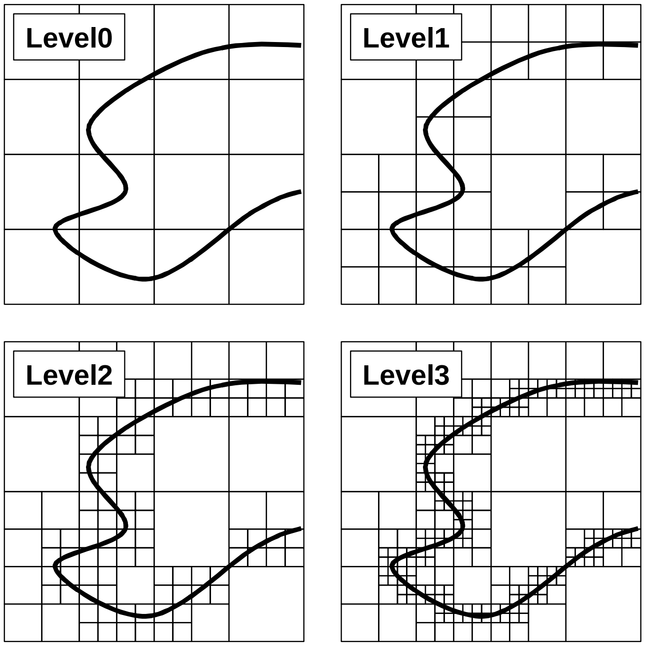


Figure 5. AMR example at multiple levels

The resulting data decomposition of the spatial grid among processors after this process remains similar to that depicted in Figure 2. However, the surface (in the X-Y plane) of each processor's sub-grid will no longer be of equal size.

Since the AMR process is data dependent, each separate calculation using SAGE will exhibit in a different adaptive refinement process and hence a different performance will result.

3. A PERFORMANCE MODEL OF SAGE

In the analysis that follows in Section 3.1, the main characteristics of SAGE as described in Sections 2.1 to 2.3 are used to construct a performance model but without AMR. This model is extended in Section 3.2 to include refinement. Applications of the model are illustrated in Section 4.

3.1 SAGE Model without AMR

The communication and computation stages of SAGE are centered around the gather/compute/scatter operations as described in Section 2.3. The runtime for one cycle of the code, given that the three stages are not overlapped, can be described as:

$$\begin{aligned} T_{cycle}(P,E) \quad = \quad & T_{comp}(E) + T_{memcon}(P,E) + \\ & T_{GScomm}(P,E) + T_{allreduce}(P) \end{aligned} \qquad (10)$$

where:

P is the number of PEs

E is the number of cells per PE

$T_{comp}(E)$ is the computation time

$T_{GScomm}(P,E)$ is the gather and scatter communication time

$T_{allreduce}(P)$ is the allreduce communication time

$T_{memcon}(P,E)$ is memory contention that may occur between PEs within an SMP box

The computation time, $T_{comp}(E)$, is measured from an execution of SAGE on a single PE for a given number of cells E.

The gather and scatter communication time is the time taken to provide boundary information by processors owning the boundary. This is related to the *PED* (the communication distance described in Section 2.2), and on the sizes of the messages. $T_{GScomm}(P,E)$ is modeled as:

$$T_{GScomm}(P,E) = C(P,E).\begin{pmatrix} 160.T_{comm}(Surface_Z.MPI_{Real8},P)+ \\ 17.T_{comm}(Surface_Z.MPI_{INT},P)+ \\ 160.T_{comm}(Surface_Y.MPI_{Real8},P)+ \\ 17.T_{comm}(Surface_Y.MPI_{INT},P)+ \\ 160.T_{comm}(Surface_X.MPI_{Real8},P)+ \\ 17.T_{comm}(Surface_X.MPI_{INT},P) \end{pmatrix} \qquad (11)$$

where $C(P,E)$ is the contention on the processor network when using P processors due to distant processor neighbor communications (i.e. $PED > 1$) and $T_{comm}(S,P)$ is the time taken to communicate a message of size S when using P processors in the system. The sizes of the messages, $Surface_Z$, $Surface_Y$, $Surface_X$ (words) are determined by the size of the sub-grid mapped to each processor, the number of processors P, and the data decomposition used as described in Section 2. For slab decomposition $Surface_Z$ = MIN$(L^2, E/2)$, $Surface_Y = 2.L$, and $surface_x = 4$ words. The size of MPI_{real8} and MPI_{INT} are determined by the MPI implementation. The coefficients multiplying the communication times in equation 11 are the frequency of the messages, as described in Section 2.3.

A linear model for the communication time is assumed which uses the Latency (L_c) and Bandwidth (B_c) of the communication network in the system. The communication latency and bandwidth vary depending on the size of the message and also the number of processors used (for instance when dealing with in-box or with out-box communication for an SMP based machine).

$$T_{comm}(S,P) = L_c(S,P) + S.\frac{1}{B_c(S,P)} \qquad (12)$$

The communication model utilizes the bandwidth and latencies of the communication network observed in a single direction when performing bi-directional communications, as is the case in SAGE. This should not be confused with the peak uni-directional communication performance of the network or peak measured bandwidths from a performance evaluation exercise (e.g. [11]).

The impact of the *PED* on communication performance depends on the specific network topology. On a cluster of Compaq ES45 SMPs, the maximum contention from an SMP box occurs when all 4 PEs within the box perform out-of-box sends and each receives from out-of-box PEs. This system's topology is a fat-tree using the Quadrics QSNet [6] as shown in Figure 6. This network is able to handle any logical PED without penalty – hence for this particular network there will be no extra overhead due to the physical distance between processors within the PED.

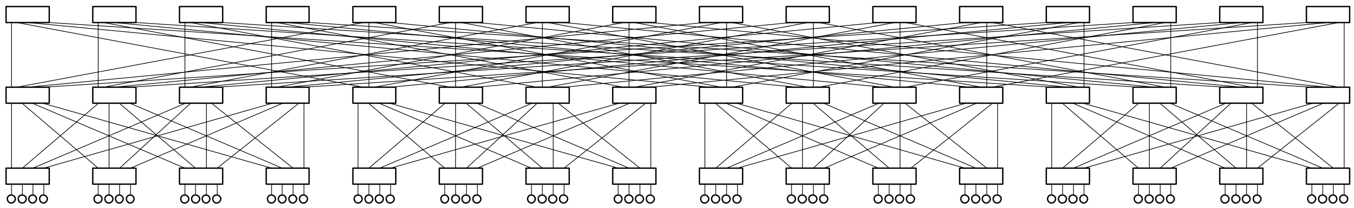


Figure 6. Network topology for a 64-node cluster of Compaq SMPs using Quadrics' QSNet Fat-tree network.

Other topologies are not contention free under this communication pattern, for example the Cray T3E, ASCI Red, and ASCI Blue Mountain. Communication involving processors within the *PED* will be bottlenecked by the lack of physical communication links between processors that limit the concurrency of messages. For example, with the initial configuration of the ASCI Blue Mountain, the minimum number of HiPPi channels that are used to interconnect SMP boxes of 128 PEs is 2, as shown in Figure 7.

The contention on the processor network, *C(P,E)*, is modeled as:

$$C(P,E) = MIN\left(MAX\left(\frac{1}{CL}\ \frac{L^2}{PEsurface}, 1 \right), \frac{P_{SMP}}{CL} \right) \tag{13}$$

where CL is the number of communication links per node, and P_{SMP} is the number of PEs per node. Thus the contention has a maximum - the number of nodes within the SMP divided by the number of communication links, i.e. when all PEs perform out-box sends and receives. It has a minimum of one since at least one PE will perform out-box communications. This model of the contention on the processor network is optimistic as it does not take into account possible overhead in the management of multiple communication links within an SMP box.

The time taken to perform the allreduce operations is modeled as:

$$T_{allreduce}(P) = 120.2.\log_2(P).T_{comm}(4,P) \tag{14}$$

which consists of log2(*P*) stages in a binary tree reduction operation. This is multiplied by 2 (since the operation is effectively a reduction followed by a broadcast). This occurs at a frequency of 120 per SAGE cycle.

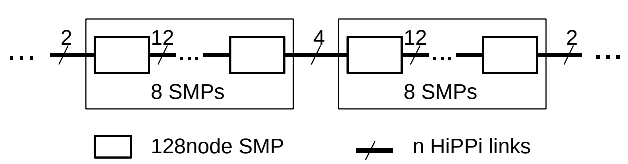


Figure 7. Inter-SMP network on ASCI Blue Mountain

The memory contention represents the extra time required per cycle when multiple PEs contend for memory within an SMP. On some systems this can be measured by considering the use of different configurations of processors for the same problem – for instance using all processors within an SMP node or using 1 processor in each of P_{SMP} nodes. The difference in execution times can be considered as the additional time due to memory contention. This is modeled as:

$$T_{memcon}(P,E) = E.T_{mem}(P) \tag{15}$$

where $T_{mem}(P)$ is the measured memory contention on P processors per cell per cycle.

Our overall model contains many inputs which may be conveniently categorized into either: application, system, or mapping parameters. These inputs specify a particular design point – a matching of the application, in a particular configuration, with a target system in a particular configuration. The application and mapping parameters can be specified appropriately for the design point based on the input deck of a specific run while the system parameters need to be measured or otherwise specified for a particular system.

The input parameters to the SAGE performance model are listed in Table 1 below.

Table 1: Input parameters to the SAGE performance model.

Application	E	Cells per processor
Mapping	$Surface_x$ $Surface_y$ $Surface_Z$	Surface size (in cells) of the sub-grid mapped to each processor (in each of the three dimensions)
System	P	Number of processors
	P_{SMP}	Processors per SMP box
	CL	Communication Links per SMP box
	$L_c(S,P)$ $B_c(S,P)$	Latencies and Bandwidths achieved in one direction on bi-directional communication
	MPI_{Real8} MPI_{INT}	Size of MPI data types
	$T_{comp}(E)$	Sequential cycle time of SAGE on E cells
	$T_{mem}(P)$	Memory contention per cell per cycle

3.2 SAGE Model with AMR

The adaptive mesh refinement process in SAGE is performed at the end of each cycle as described in Section 2.4. The AMR operation is triggered by one of several thresholds on the physical quantities contained in the cell. Thus it is heavily dependent on the calculation being performed. In order to accurately model this operation information on the AMR process due to the calculation being performed is required. This includes:

- the number of new cells added in a cycle,
- the current cell division factor (the total cells divided by the number of cells level 0 cells), and
- movement of cells between processors to load-balance.

For a particular calculation this information needs to be defined on a per-cycle basis. For example, a calculation which results in intense physical phenomena in a localized area of the spatial grid will require more time to load-balance (see Section 2.4) in contrast with a calculation with uniformly distributed phenomena.

The performance model of SAGE presented in Section 3 can be extended to include the main characteristics of the adaptive mesh refinement process. A model that includes these operations is:

$$\begin{aligned} T_{cycle_i}(P, E, \mathbf{D}, \mathbf{A}, \mathbf{M}_{\mathbf{cm}}) = \; & T_{comp}(E.D_i) + T_{memcon}(P, E.D_i) + \\ & T_{allreduce}(P) + T_{GScomm}(P, E, D_i) + \\ & T_{divide}(A_i) + T_{combine}(E.D_i) + \\ & T_{load}(M_{cm_i}, P) \end{aligned} \tag{16}$$

The main additional components in this model in comparison to that defined previously in equation 10, are:

$T_{divide}(A_i)$ - the time to divide cells in the current cycle

$T_{combine}(E.D_i)$ - the time to combine cells in the current cycle

$T_{load}(M_{cmi}, P)$ - the time to perform the load-balancing

In addition three parameters are included in this model that define characteristics of the calculation being performed. Each represents a time history of values defined on a cycle by cycle basis:

D - a vector containing the cell division factor $[1..8^{maxlevel}]$

A - a vector containing the maximum number of cells added (over all processors) through the division process

$\mathbf{M}_{\mathbf{cm}}$ - a vector containing the maximum number of cells moved between any two PEs in the load balancing.

By defining these inputs on a per cycle basis, the model can accurately encompass the change in computation time and communication time due to the change in the amount of cell division. For instance, the computation time will scale in proportion to the amount of cell division (the volume of the sub-grid) whereas the size of communications for the gather and scatter operations will scale as the 2/3 power of the cell division (the surface of the sub-grid). This model is not described in any further detail in this work. An application of the model with adaption is illustrated in Section 4.3.

Table 2. System parameters used in the validation for each system.

System	AlphaSever ES45	AlphaServer ES40	ASCI Blue Mountain	ASCI White
P_{smp} (PEs per node)	4	4	128	16
# Nodes used in Validation	8	64	40	256
CL (Communication Links per node)	1	1	$\begin{cases} 8 & P \le 1024 \\ 4 & 1024 < P \le 2048 \\ 2 & P > 2048 \end{cases}$	2
$T_{comp}(E)$ (s)	0.36	0.48	1.80	0.77
$L_c(S,P)$ (µs)	*in box*$(P \le 4)$ $\begin{cases} 4.8 & S < 64 \\ 4.9 & 64 \le S \le 256 \\ 13.5 & 256 < S \le 8192 \\ 23.2 & S > 8192 \end{cases}$ *out box*$(P > 4)$ $\begin{cases} 6.10 & S < 64 \\ 6.44 & 64 \le S \le 512 \\ 13.8 & S > 512 \end{cases}$	*in box*$(P \le 4)$ $\begin{cases} 12.7 & S < 64 \\ 12.8 & 64 \le S \le 256 \\ 30.3 & 256 < S \le 8192 \\ 25.7 & S > 8192 \end{cases}$ *out box*$(P > 4)$ $\begin{cases} 9.28 & S < 64 \\ 9.00 & 64 \le S \le 512 \\ 21.4 & S > 512 \end{cases}$	*in box*$(P \le 128)$ $\begin{cases} 8.0 & S < 128 \\ 11.5 & 128 \le S \le 512 \\ 15.0 & 512 < S \le 2048 \\ 18.7 & S > 2048 \end{cases}$ *out box*$(P > 128)$ $150 \quad \forall S$	*in box*$(P \le 16)$ $\begin{cases} 12.0 & S \le 128 \\ 17.0 & 128 < S \le 512 \\ 19.0 & S > 512 \end{cases}$ *out box*$(P > 16)$ $\begin{cases} 18.0 & S \le 128 \\ 25.0 & 128 < S \le 4096 \\ 87.0 & 4096 < S \le 65536 \\ 28.3 & S > 65536 \end{cases}$
$1/B_c(S,P)$ (ns)	*in box*$(P \le 4)$ $\begin{cases} 0.0 & S < 64 \\ 13.9 & 64 \le S \le 256 \\ 1.04 & 256 < S \le 8192 \\ 1.37 & S > 8192 \end{cases}$ *out box*$(P > 4)$ $\begin{cases} 0.0 & S < 64 \\ 12.2 & 64 \le S \le 512 \\ 8.30 & S > 512 \end{cases}$	*in box*$(P \le 4)$ $\begin{cases} 0.0 & S < 64 \\ 24 & 64 \le S \le 256 \\ 9.0 & 256 < S \le 8192 \\ 3.2 & S > 8192 \end{cases}$ *out box*$(P > 4)$ $\begin{cases} 0.0 & S < 64 \\ 25.5 & 64 \le S \le 512 \\ 13.7 & S > 512 \end{cases}$	*in box*$(P \le 128)$ $\begin{cases} 0.0 & S < 128 \\ 13.9 & 128 \le S \le 512 \\ 7.4 & 512 < S \le 2048 \\ 7.3 & S > 2048 \end{cases}$ *out box*$(P > 128)$ $10 \quad \forall S$	*in box*$(P \le 16)$ $\begin{cases} 21.6 & S \le 128 \\ 2.4 & 128 < S \le 512 \\ 2.0 & S > 512 \end{cases}$ *out box*$(P > 16)$ $\begin{cases} 84.6 & S \le 128 \\ 16.6 & 128 < S \le 4096 \\ 8.46 & 4096 < S \le 65536 \\ 4.32 & S > 65536 \end{cases}$
$T_{mem}(P)$ (µs)	$\begin{cases} 1.8 & P = 2 \\ 4.8 & P > 2 \end{cases}$	$\begin{cases} 2.2 & P = 2 \\ 4.4 & P > 2 \end{cases}$	-	-

4. APPLICATION OF THE MODEL

In this section the SAGE performance model described in Section 3 is validated against four existing architectures and also applied to predicting performance on future architectures. In addition, the performance of SAGE is investigated given algorithmic changes that could be implemented in the code. An example of predicting the performance of SAGE with the adaptive mesh refinement process is also illustrated.

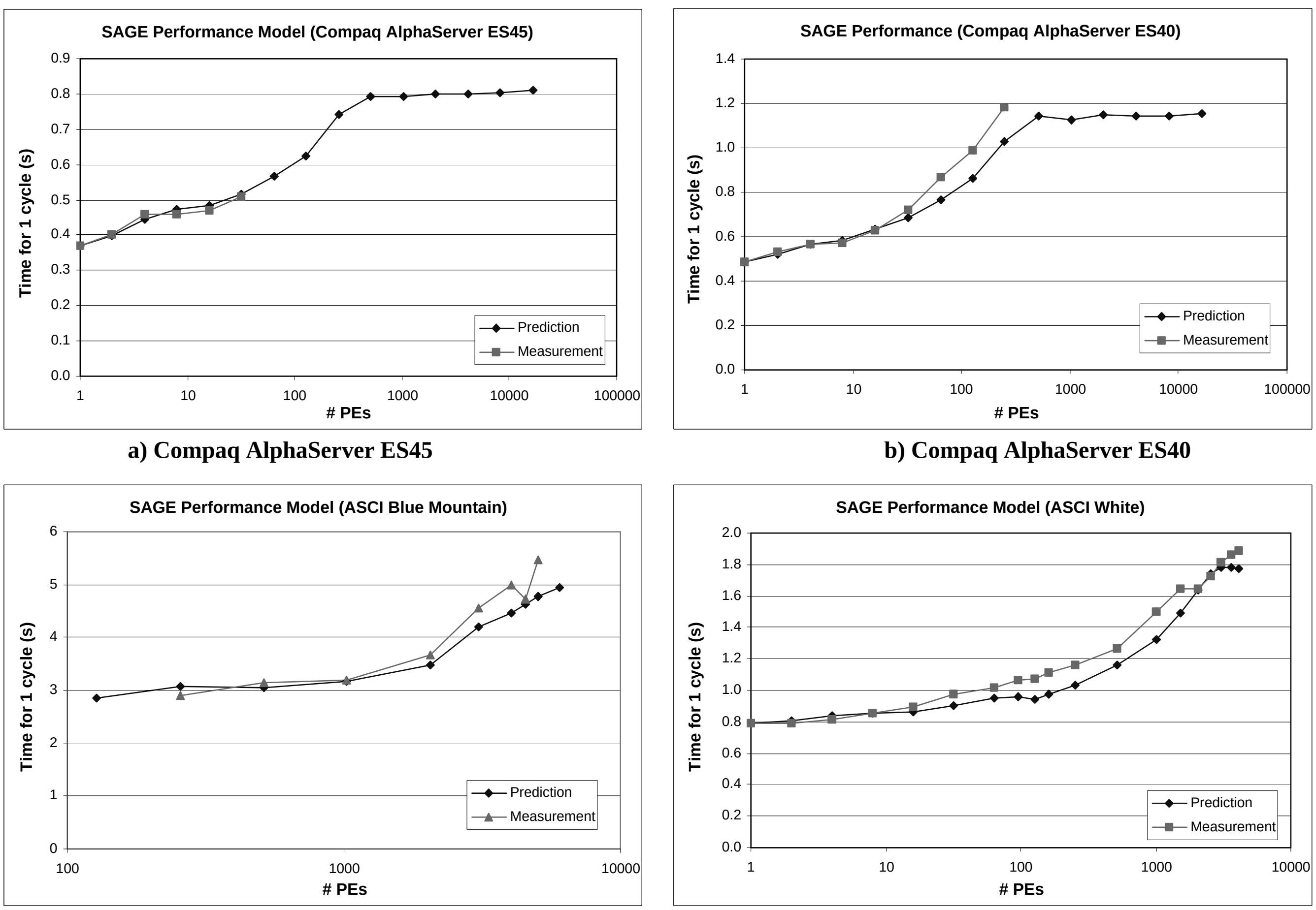
4.1. Validation and Performance Prediction on Future Architectures

The model presented in Section 3 has been validated against measurements taken on a Compaq AlphaServer ES45, an AlphaServer ES40, the ASCI Blue Mountain (SGI Origin 2000), and from a preliminary performance analysis of ASCI White (IBM SP3). A detailed performance study of ASCI White can be found in [8]. The input parameters to the SAGE performance model are listed in Table 2. This includes the number of nodes used in the validation along with the system parameters used in the model for each architecture. The parameters are either measured or otherwise specified. The comparison is performed with the default slab decomposition in SAGE as described in Section 2.

For each system, the time taken to perform one cycle of SAGE as given by the performance model is compared to measurements. In the case of the two Compaq systems, predictions are given for systems larger than was available in the measurement process. The Compaq AlphaServer ES45 cluster that we had access to was very small, but a larger system of this kind is being installed at Los Alamos National Laboratory. Without an available large-scale system, the performance model is able to provide an expectation of the performance on such a future architecture.

The comparison of predictions and measurements on the four systems is shown in Figure 8. The predictions from the performance model show high accuracy – mostly within 10% of the actual measurements.

A comparison of the cell-cycles per second on SAGE is shown in Figure 9. This metric is used by SAGE as a further indication of performance. It represents the number of cells that can be processed in each wall-clock time unit. In Figure 9 measurements are used for the CRAY T3E, ASCI White, and ASCI Blue Mountain, whereas we predict the performance of the Compaq system using our model.



a) Compaq AlphaServer ES45

b) Compaq AlphaServer ES40

c) ASCI Blue Mountain (SGI Origin 2000)

d) ASCI White (IBM SP3)

Figure 8. Comparison of predictions from the SAGE performance model with actual measurements

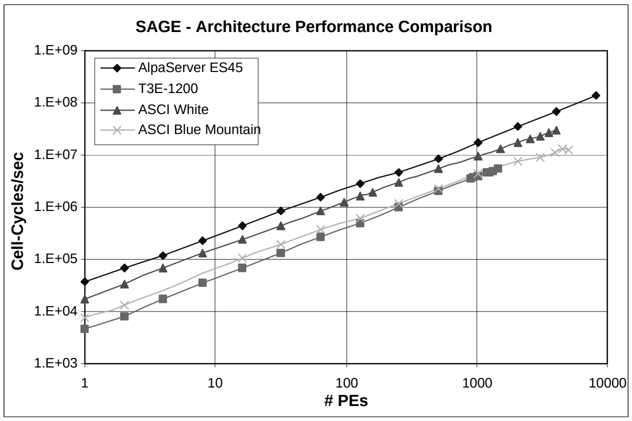


Figure 9. Comparison of the performance of SAGE across several systems.

The model predicts the performance of the AlphaServer ES45 to be approximately a factor of 2 times greater than that on the ASCI White (IBM SP3) on a comparable number of processors. A system with a peak performance of 30Tflops composed of the Compaq SMP boxes with Quadrics QSNet would be approximately 20 times greater than the performance of SAGE achieved to date on the ASCI Blue Mountain with 6000 SGI Origin 2000 processors. By comparison, the ratio of peak speeds is approximately 10.

4.2. Performance Prediction on Algorithmic Transformations: an alternative Data Decomposition

The surface-to-volume ratio of the processing in SAGE is dependent on the grid decomposition. There is a large difference between the use of the slab decomposition (Figure 1) and a "cube" decomposition (Figure 10). Where the slab decomposition results in communications scaling as the 2/3 power of the number of PEs, as shown by equation (3), with a cube decomposition the communication size will remain approximately constant, though the number of PE pairs communicating will be larger. It can be easily shown (see for example [2]) that the surface-to-volume ratio (i.e. the communication-to-computation ratio) gets better (i.e. smaller) as the aspect ratio of the sub-grids changes towards being perfect cubes, as suggested in Figure 10. Of course, perfect cubic decomposition can only be achieved when the number of processors is a cubic power, as is the decomposition on 8 processors shown in Figure 10.

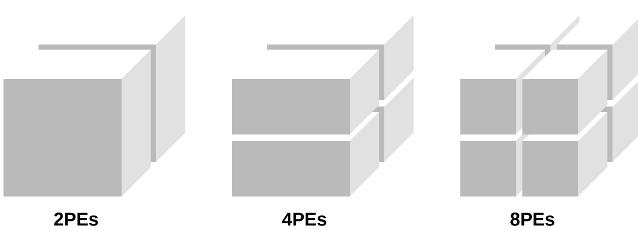
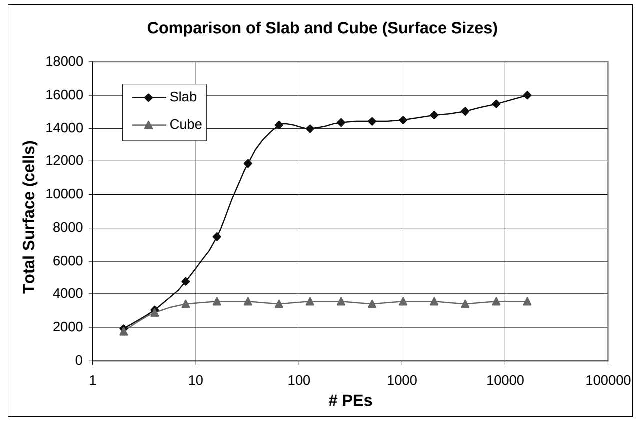
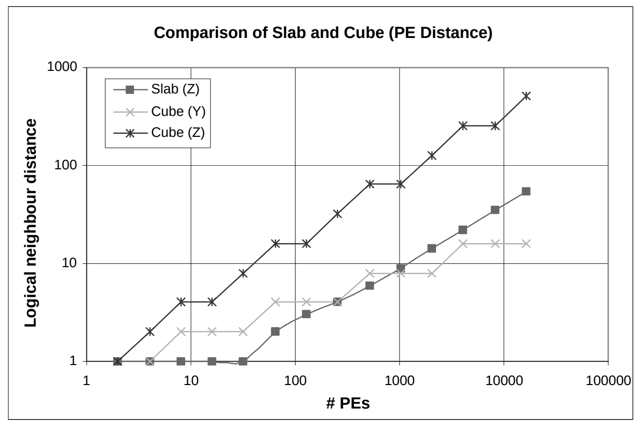


Figure 10. Possible 3-D data decomposition configurations for 2, 4 and 8 processors



a) Surface sizes

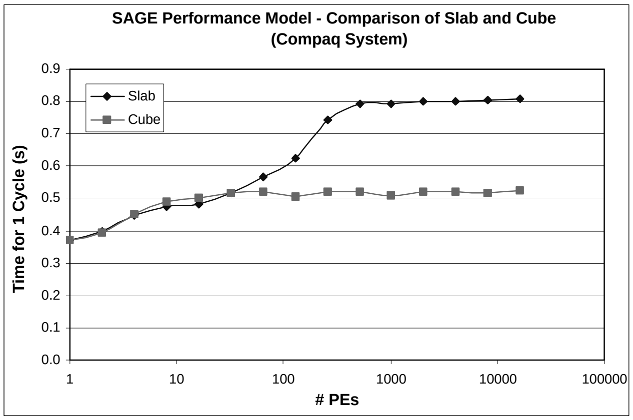


b) PE distance (PED)

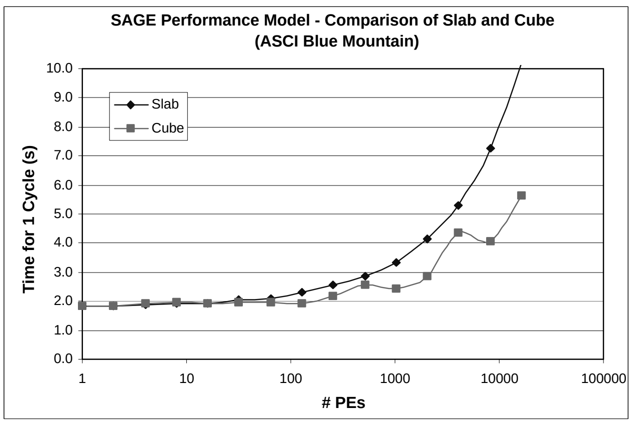
Figure 11. Comparison of Slab and Cube decomposition

A comparison between the cube decomposition and the slab decomposition is shown in Figure 11. The total surface of the sub-grid on an individual PE is plotted which is proportional to the communication that takes place in each gather (and scatter) operation. The PE distance (*PED*) is also shown in Figure 11b). The curves for the slab decomposition have already been presented in Figure 2. The *PED* for the slab decomposition in the X and Y dimensions are always equal to 1. For the cube decomposition PED is always equal to 1 in the X dimension, but varies in the Y and Z dimensions.

The communication size using the cube decomposition is considerably smaller than that for the slab, but the PED is considerably larger. A comparison between the expected performance of SAGE using cube decomposition and the current slab decomposition on the Compaq AlphaServer ES45, and the ASCI Blue Mountain, is shown in Figure 12. This is achieved by modifying the parameters $Surface_X$, $Surface_Y$, and $Surface_Z$, in equation 11, to represent the sub-grid surface sizes in the cube decomposition. For an ideal cube these would all be equal to $\left(L / P^{1/3}\right)^2$. The use of the cube decomposition reduces communication requirements and hence results in an expected performance improvement of 35% (on the Compaq system), and between 15% and 45% (on the SGI system) compared with the use of slabs.

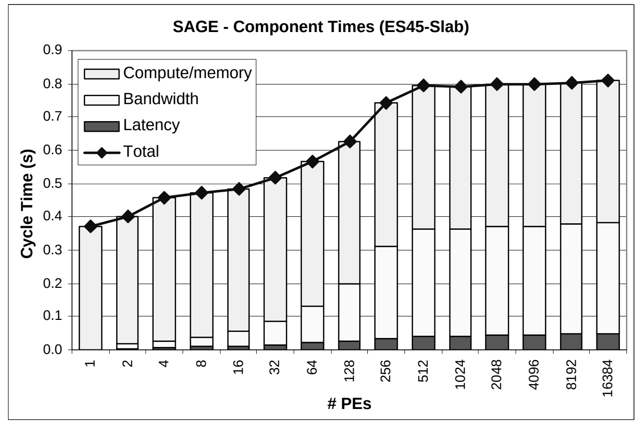


a) Compaq AlphaServer ES45

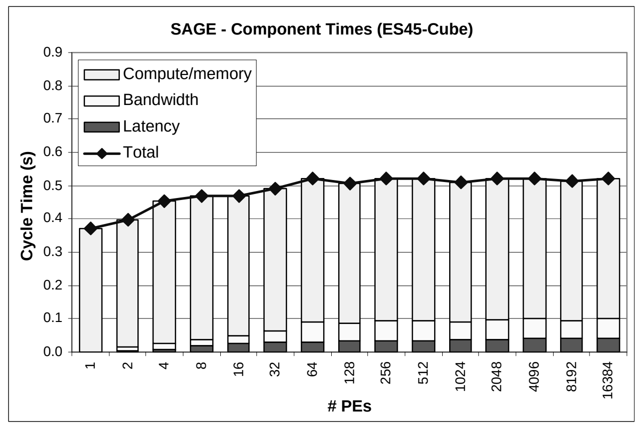


b) ASCI Blue Mountain

Figure 12. Performance comparison of slab and cube decomposition



a) Slab decomposition



b) Cube decomposition

Figure 13. Time-component predictions (Compaq ES45)

The performance model can also be used to provide insight into where time is spent within the application. In Figure 13, time components representing computation (including memory), communication latency, and communication bandwidth are shown for both data decomposition schemes of SAGE on the Compaq AlphaServer ES45. It can be clearly seen that the communication bandwidth component is much reduced when using the cube decomposition. Both the computation and the communication latency components remain mostly unchanged.

SAGE could benefit from a cube decomposition of the full grid if the communication network within the machine is able to handle the large logical *PEDs* without performance penalty. This is true in the fat-tree topology of the Quadrics network used on the cluster of Compaq SMPs as described in Section 3.

4.3 Extending the Performance Model to AMR

The model can be used to explore the performance on different characteristic adaptive mesh refinement calculations on different architectures. In Figure 14 an example time history for each of the cell division factors (**D**), the maximum cells added in a cycle (**A**), and the maximum cells moved for load balancing ($\mathbf{M_{cm}}$) are shown. The example time histories attempt to depict the situation in which a shock-wave propagates through the spatial grid. This causes the following characteristics:

- the cell division factor gradually increases with the number of cells added per cycle as the shock-wave expands thus the cell division factor increases, and
- load-balancing is assumed to take place every fifth cycle.

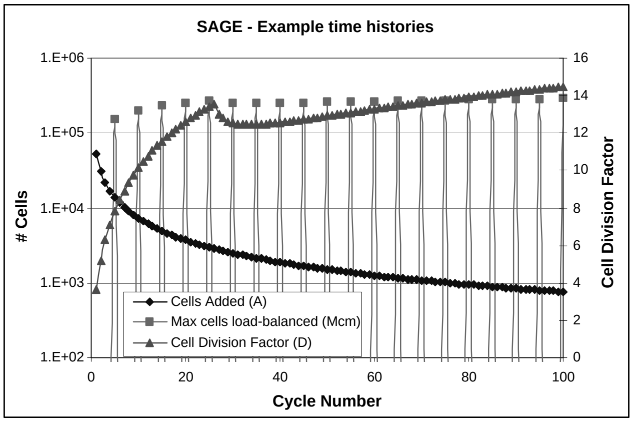
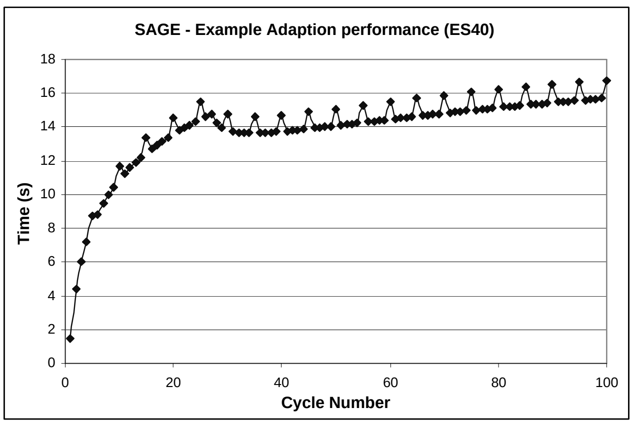
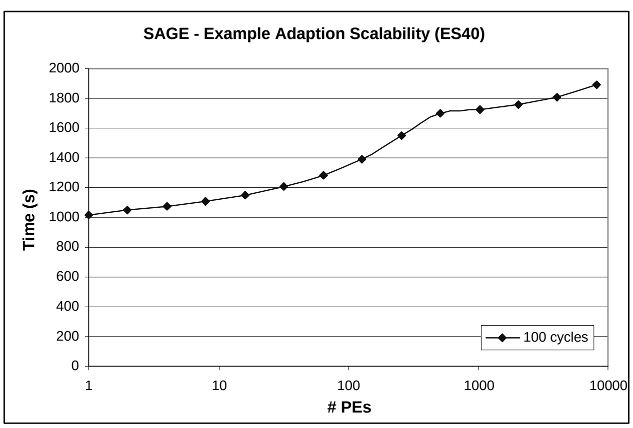


Figure 14. Example time histories for: division factor, added blocks, and blocks load-balanced (indexed by cycle).

The time histories as depicted in Figure 14 were used in the performance model in order to investigate both the variation in cycle time during the calculation (Figure 15a), and the time taken to perform the 100 cycles while scaling the number of processors (Figure 15b). This was undertaken for the Compaq AlphaServer ES40. It can be seen from Figure 15a), that cycles requiring load-balancing take slightly longer than those without.



a) time taken for each of 100 cycles



b) Example adaption scalabilty (time for 100 cycles)

Figure 15. Performance Prediction of SAGE with AMR (using the input histories from Figure 14)

5. SUMMARY AND CONCLUSIONS

In this paper we have presented a predictive performance and scalability model for an important application from the ASCI workload. The model takes into account the main computation and communication characteristics of the entire code. The model proposed was validated on two large -scale ASCI architectures, ASCI White (IBM SP3), and ASCI Blue Mountain (SGI Origin 2000), showing very good accuracy. The model was then utilized to predict performance of SAGE on future architectures and also when using an alternative parallel data decomposition.

We believe that performance modeling is the key to building performance engineered applications and architectures. To this end, the work presented in this paper represents one of a very few existing perfo rmance models of entire applications. Like our previous performance model of a particle transport application [4], the model incorporates information from various levels of the benchmark hierarchy [3] and is parametric - basic machine performance numbers (latency, computational rate, bandwidth) and application characteristics (problem size, decomposition method, etc.) serve as input. Such a model adds insight into the performance of current systems, revealing bottlenecks and showing where tuning efforts wo uld be most effective. It also allows prediction of performance on future systems. The latter is important for both application and system architecture design as well as for the procurement of supercomputer architectures

A performance model is meant to be *updated*, *refined*, and *further validated* as new factors come into play. The work performed in this report was primarily concerned with the analysis of SAGE in absence of grid adaptation. With additional analysis, the model has been extended to include the main characteristics of the adaptation process. The performance model can be used to investigate performance on alternative application configurations (data decompositions), and alternative target systems.

ACKNOWLEDGEMENTS

This work was supported by fundi ng from the Los Alamos Computer Science Institute. We would like to thank Ed Benson for access and support on the AlphaServer ES45 at Compaq in Marlborough, MA. Los Alamos National Laboratory is operated by the University of California for the National Nuc lear Security Administration of the US Department of Energy.

6. REFERENCES

[1] Culler, D.E., Singh, J.P., Gupta, A., Parallel Computer Architecture, Morgan Kaufmann, ISBN 1-55860-343-3, 1999.

[2] Goedecker, S., Hoisie, A., Performance Optimization of Numerically Intensive Codes, SIAM Press, ISBN 0-89871-484-2, March 2001.

[3] Hockney, R., Berry, M., (Eds). Public International Benchmarks for Parallel Computers. Scientific Programming, Vol. 3, 1994, 101-104.

[4] Hoisie. A., Lubeck, O., Wasserman. H., Performance and scalability analysis of teraflop-scale parallel architectures using multidimensional wavefront applications, Int. J. of High Performance Computing Applications, Vol. 14, No. 4, Winter 200, 330-346.

[5] Nudd, G.R., Kerbyson, D.J., et.al. PACE: A Toolset for the Performance Prediction of Parallel and Distributed Systems, in the Journal of High Performance Applications, Vol. 14, No. 3, Fall 2000, 228-251.

[6] Petrini, F., Feng, W., Hoisie, A., Coll, S., and Frachtenberg, E. The Quadrics Network (QsNet): High-Performance Clustering Technology, Symposium on High Performance Interconnects, Hot Interconnects 9, Stanford CA, August 22 - 24, 2001.

[7] Rauber, T., and Runger, G., Modeling the Runtime of Scientific Programs on Parallel Computers, in Proc. 2000 ICPP Workshops. IEEE Computer Society, 2000, 307-314.

[8] de Supinski, B.R. The ASCI PSE Milepost: Run-Time Systems Performance Tests, Int. Conf. On Parallel & Distrib. Process. Tech. & Apps., Las Vegas, June 25-28, Vol. 4, 2001, 1987-1993.

[9] Weaver, R., Major 3-D Parallel Simulations, BITS - Computing and communication news, Los Alamos National Laboratory, June/July, 1999, 9-11. http://www.lanl.gov/orgs/cic/cic6/bits/99june_julybits/opener.html

[10] Worley. P.H., Performance Tuning and Evaluation of a Parallel Community Climate Model, SC99, Portland, Oregon, November 1999.

[11] Worley, P.H. Performance Evaluation of the IBM SP and the Compaq AlphaServer SC. In Proc. ICS 2000, ACM, 2000, 235-244.